1425

NPS52-89-020

NAVAL POSTGRADUATE SCHOOL

Monterey, California

A MULTIMEDIA DATABASE MANAGEMENT SYSTEM SUPPORTING CONTENTS SEARCH IN MEDIA DATA

Vincent Y. Lum
Klaus Meyer-Wegener

March 1989

Approved for Public Release; distribution is unlimited.

Prepared for:

Naval Postgraduate School
Monterey, CA 93943

FEDDOCS
D 208.14/2
NPS-52-89-020

GORDON KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA 93943

NPS52-89-020

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rear Admiral R. C. Austin
Superintendent

Harrison Shull
Provost

This report was prepared in conjunction with research conducted for the Naval Ocean Systems Center and funded by the Naval Postgraduate School.

Reproduction of all or part of this report is authorized.

This report was prepared by:

MONTEREY, CALIFORNIA

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NPS52-89-020	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL (*If applicable*)	7a. NAME OF MONITORING ORGANIZATION Naval Ocean Systems Center
6c. ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943	7b. ADDRESS (*City, State, and ZIP Code*) San Diego, CA 92152
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Naval Postgraduate School 8b OFFICE SYMBOL (*If applicable*)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER O&MN, Direct Funding
8c. ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO.

11. TITLE (*Include Security Classification*)
A MULTIMEDIA DATABASE MANAGEMENT SYSTEM SUPPORTING CONTENTS SEARCH IN MEDIA DATA (U)

12. PERSONAL AUTHOR(S)
Lum, Vincent Y., Meyer-Wegener, Klaus

13a. TYPE OF REPORT	13b TIME COVERED	14. DATE OF REPORT (*Year, Month, Day*)	15 PAGE COUNT
Progress	FROM Oct 88 TO Jan 89	1989 March	33

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18 SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUB-GROUP	Multimedia database systems, content search, natural language descriptions, knowledge representation, database architecture

19 ABSTRACT (*Continue on reverse if necessary and identify by block number*)
It is feasible now to store and manage in computers new types of data like text, images, graphics, and sound recordings. This paper claims that database management systems should be extended to organize these new types of data and to enable search based on their contents. Media objects are modelled as attributes of abstract data types. The contents is captured in terms of natural language descriptions and is translated by a parser into predicated for easy matching with query phrases. The implications of this approach are discussed: data organization for multimedia including contents representation, the dictionary used by the parser to recognize words and assign predicates, rules to utilize semantic relationships in the query evaluation, and access paths to speed up the search for the descriptions. The result is an architecture for multimedia database management systems that combines the additional components needed with the conventional data management and identifies their ways of interaction. Three types of user interfaces are offered that require different levels of skill. The architecture is modular and allows to integrate more advanced AI techniques once they become available.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Vincent Y. Lum	22b TELEPHONE (*Include Area Code*) (408) 646-2693 22c. OFFICE SYMBOL 52Lu

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted.
All other editions are obsolete

☆ U.S. Government Printing Office: 1986—606-24.

A Multimedia Database Management System

Supporting Contents Search in Media Data

Vincent Y. Lum, Klaus Meyer-Wegener

Naval Postgraduate School
Department of Computer Science
Code 52
Monterey, CA 93943
U.S.A.
Phone: (408) 646-2449
E-mail: lum@cs.nps.navy.mil

Abstract

It is feasible now to store and manage in computers new types of data like text, images, graphics, and sound recordings. This paper claims that database management systems should be extended to organize these new types of data and to enable search based on their contents. Media objects are modelled as attributes of abstract data types. The contents is captured in terms of natural language descriptions and is translated by a parser into predicates for easy matching with query phrases. The implications of this approach are discussed: data organization for multimedia including contents representation, the dictionary used by the parser to recognize words and assign predicates, rules to utilize semantic relationships in the query evaluation, and access paths to speed up the search for the descriptions. The result is an architecture for multimedia database management systems that combines the additional components needed with the conventional data management and identifies their ways of interaction. Three types of user interfaces are offered that require different levels of skill. The architecture is modular and allows to integrate more advanced AI techniques once they become available.

Klaus Meyer-Wegener's permanent address is: Universitaet Kaiserslautern, Fachbereich Informatik, Postfach 30 49, 6750 Kaiserslautern, West Germany, e-mail: meyerweg@uklirb.uucp.

Direct Funding (with external "sponsor")

Keywords

Multimedia database systems, content search, natural language descriptions, knowledge representation, database architecture

1. Introduction

In addition to alphanumeric and text data, technology today has made it possible to capture and store in computers image, sound, signal and other media data generally referred to as multimedia data or simply media data. Indeed many applications routinely need data of these kinds. Such are the cases of military applications, publishing applications, or instructional applications. Current technology allows us to keep these different media data in separate files. That is, a single image or a single signal, which we shall call a "media object" in this paper, will generate one distinct file. Although the term object is used, it is obvious that a media object is merely a value of an instance in multimedia data. That is, as in normal database in which age is an attribute and the value is 35, in multimedia data an image is an object but is also the value of the attribute picture. It does not require much imagination to see that under this circumstance, a user would soon lose track of the "objects", even for a very small application.

Handling data means allowing to search and store by the *content* of the data we process. Immediately, one would then ask the question of how to handle content search in multimedia data. There is no question that this is a difficult problem. One must find ways of handling a very large amount of multimedia data and of having the capability of searching and finding the appropriate data conveniently and efficiently based on the contents of the multimedia data that are frequently complex to define.

A similar problem, though in a much simpler form, was encountered earlier when dealing with the more "standard" types of data, i.e. alphanumeric and textual data. Database management systems (DBMS) were developed as a result. The power of DBMS is well recognized today and there is no need for us to discuss it here. What one wishes to develop is a technology that would allow us to handle multimedia data as conveniently as we can process the standard data. In recent years efforts are being made to extend the concepts of DBMS to fulfil this goal. *Multimedia Database Management Systems (MDBMS)* [WK87, LM88] research has thus been born.

The use of multimedia data is actually a natural extension of the use of the computer to process standard data. For example, in a persons database, one would store the various data such as birthday, address, job title, etc., about an individual. In many applications one would most likely want to store the photo of the individual as well, if we have a technology to do this easily. Storage and processing requirements make such routine application still not so

simple. Nevertheless we are getting very close to be able to do so from the technology's standpoint.

A number of projects have been established to do research in multimedia data processing. Among these include the MINOS project at the University of Waterloo [Ch86], which is aimed at the management of documents containing multimedia data, and the ORION system in MCC in Austin, that contains a Multimedia Information Manager (MIM) [WK87] for processing multimedia data. The IBM Tokyo Research Laboratory has developed two "mixed-object database systems," MODES1 and MODES2 [KKS87], and in Europe there is an ESPRIT project designing a multimedia filing system called MULTOS [Be85, Be86, BRG88]. Today, we are still in the infancy stage in this area of research. Even the definition of the functionality of MDBMS is still an open issue.

Further, although many researchers pursue to develop multimedia database management systems, and some of them have taken a revolutionary approach of developing an object-oriented system, none of them has put much of an effort to address the problem of content search of multimedia data. This problem in our view is believed to be of paramount importance in handling multimedia data. It also happens to be a difficult problem to solve, because multimedia data, unlike standard data, is very rich in semantics for each data instance, e.g. an image. In the traditional standard data, a number 10 for an attribute "balance" in the bank deposit database can mean only very few things; but a simple image associated with a person's attribute "photo" has a great deal more information implicitly included with the image, e.g. blonde hair, scar on the chin, etc. Without the ability to do content search, a multimedia database management system would be of very little use. It would be like having a database management system that cannot process query matches.

Consider an example that one uses the computer to store newspapers. Moreover, one may even consider the storage to include sound and video recordings of speeches, debates, etc. How are we to find photos and recordings containing Reagan and Gorbachev signing a treaty? How are we to find photos and recordings of snow avalanches burying villages? Pattern matching as frequently done in textual search is not useful. Exact matching as is normally done in DBMS is totally inadequate.

Our project in the Naval Postgraduate School has been initiated to address the problem of handling multimedia data with the emphasis of providing content search capability. Our approach is to integrate the artificial intelligence, DBMS and abstract data type techniques to develop a system that will allow us to process multimedia data similar to the way standard data are handled. How do we accomplish this? We make use of the method in which people record history through textual descriptions. Thus with each media data object, a *description* of noun phrases and even complete sentences is included. For example, "Reagan and

Gorbachev signing a treaty" and "the event occurred on December 7, 1987" would accompany the photos or recordings associated with this event. The handling of the multimedia data will be done with the use of the *abstract data type concept*. Image, sound, signal, text, and graphic data will be treated as new data types [MLW88]. Any attribute can have one of these types instead of the usual integer or char. The normal query language will be used to retrieve or update the value, present it on output devices, or overwrite it with newly captured data. The operations defined for the abstract data type will be used to process the multimedia data "values". The definition of the operations for these data types is one of the tasks for developing a MDBMS.

To be sure, the use of text description for multimedia data alone will not solve the problem of content search of multimedia data. Information retrieval in the library science has attempted to solve the problem of retrieving books by contents for many years. We need ways of narrowing the meaning of content search and getting us to a better focus. To do this we will employ artificial intelligence techniques as discussed later. In addition, to be able to support searches efficiently, additional data structures other than the normal ones in a DBMS are needed in many parts of the system. They will be discussed in a later section when we present the various components in the system.

To our knowledge the broad capability approach of using natural language description and AI techniques to process multimedia data is the first proposal of its kind. In fact, as we stated, research today in media data has not even addressed the problem of content search. Further, the investigation and the definition of the abstract data type operations for multimedia data though not new is definitely still an open research problem. Exactly what operations are to be defined for what kind of media data has not been specified. As we will see, this approach blends naturally with the existing DBMS techniques.

In the following sections, we shall expand first a little more on the use of descriptions to enhance the multimedia data for processing purposes. In Section 3 we shall describe how the descriptions are converted into predicates and how we avoid the difficulties of the imprecisions, ambiguities and subjectivities associated with any natural language. In Section 4 the extension of a data model to include abstract data types for handling multimedia data and the definition of the operations are proposed and in 5 a discussion of the creation of additional data structures to support such a system is included. Finally the paper will outline an architecture on which the system is being implemented and a discussion of the future work will be presented in the conclusion.

2. Data Organization for Multimedia Data

As stated in the introduction multimedia data is rich in sematics and much information is implicitly defined, making it impossible to do content search, if no additional help is available. Our proposal calls for the attachment of a textual description, named *description data,* to accompany each media object. For example, the description "Reagan and Gorbachev signing a treaty", "the event occurred on December 7, 1987" and others like "it took place in Washington, D.C." will be included in the description data for the appropriate photos and recordings. Frequently this description data is per se redundant as it repeats the information contained in the media data like an image. However, the complexity and depth of a media object's information content makes the problem of content search of the media data practically impossible without the description data.

Description data is generally difficult to obtain. Automatic derivation by the computer is deemed to be normally impossible in today's technology. It is therefore proposed that users will supply the description data for the media data in *natural language form.* While this necessitates the interpretation of media data to become subjective, such an approach has a long history of acceptance in many applications. For example, abstractions and summaries are routinely requested from authors by the publishers, interpretations of graphs and radars are done by technicians, and X-ray photos are interpreted by doctors to turn into textual descriptions.

A description of a media object can be fairly rich and complicated, or it can be simple, depending on the application. The decision is completely left to the users and the system will not dictate it one way or another. However, the system will provide the necessary software to support its applications in terms of search, creation, or modification as discussed later.

Description data is not the only data that we store with the media data. Intrinsic to the storage of media data is the *registration data.* Registration data generally enhances the information about the media data and is not redundant. This part of the data usually tells us a different aspect about the multimedia data that is not included in the *raw data* which is the bit string representation of the image, sound, signal, etc. obtained from the process of digitizing the original multimedia object. For example, an image may have been converted into bit strings for the computer to store. However, in order for us to process the data, we must have information like resolution, pixels depth, source, colormap, etc. We may also want to have information such as the date of capture and the instrument used for capturing it, etc. The important issue about the registration data is that it is required if anything is to be done with the data, either to interprete or display them, indentify or distinguish them from others, and so on.

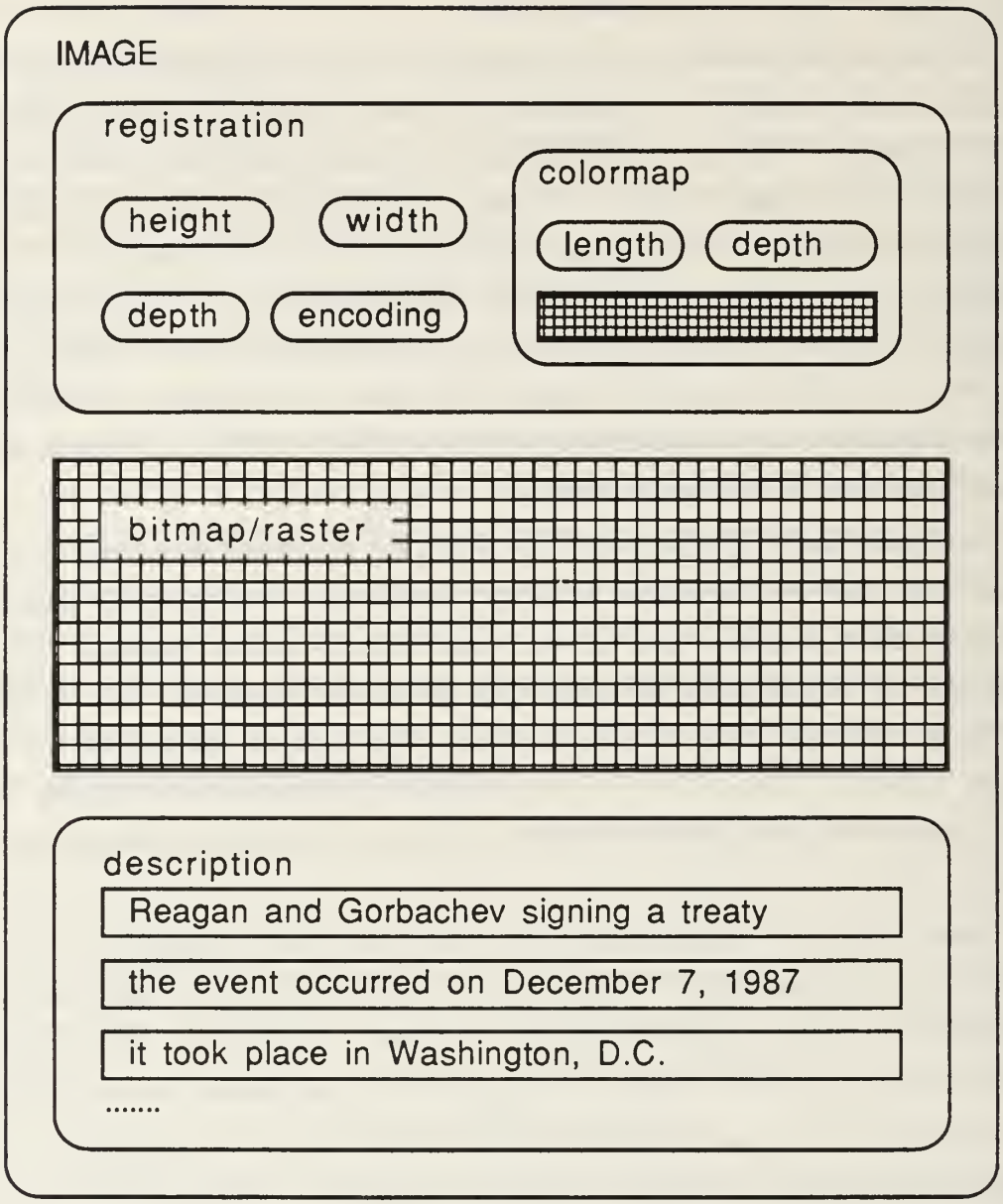

Figure 1: Internal Structure of a Media Object (using image as an example)

In our scheme, then, each media object will provide us with three parts as shown in Figure 1: raw data, registration data, and description data. (Although Figure 1 relates only to image data, other kind of multimedia data follow the same pattern.) The use of the different parts of these data is given in Figure 2. Naturally, *operations* must be provided to access and process these data. These operations will be discussed in Section 4 when we talked about employing the abstract data type concept to handle multimedia data. However, one should be

able to see that, in order to access the raw data, one must go "through" the registration data. One must also provide some editing operations for the raw data as well. Editing operations on the raw data including filtering and clipping may be permissible. Special operations on the raw data and description may have to be defined. For example, a user may want to edit the colormap to see the effect of different colors on the images. Such editing may be useful. Processing the data on content will be entirely done on the description data. In fact, it is expected that most of the processing done by the MDBMS will not touch the raw data. As we shall see later in Section 4, many of these operations will be done through the commands of the query language.

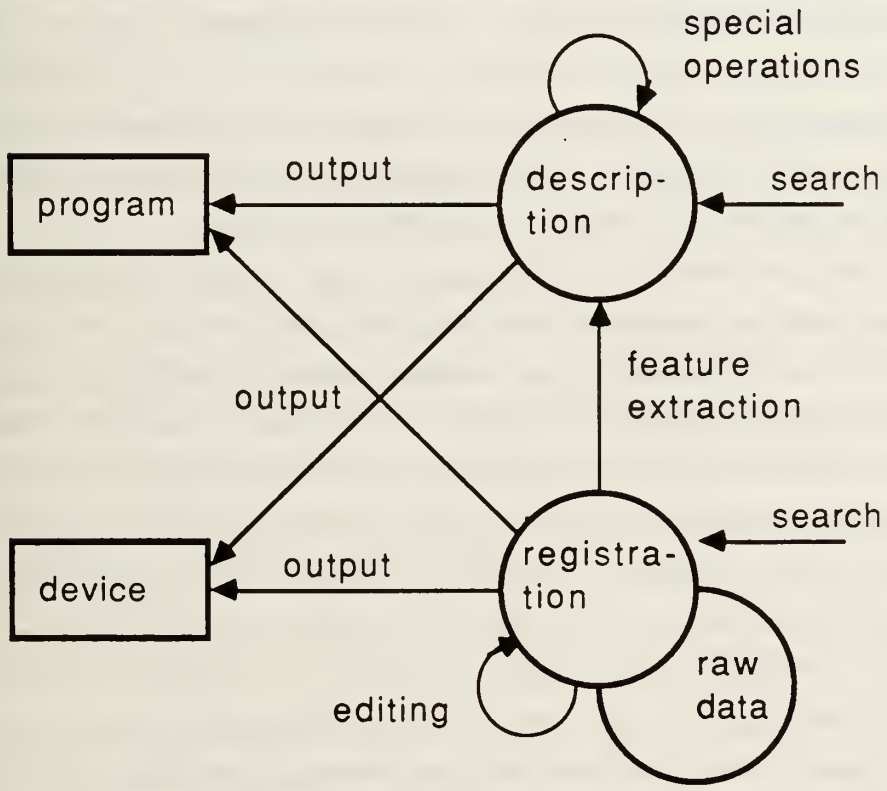

Figure 2: Operations on the Components of Media Objects

3. Issues in Content Representation

As mentioned before natural language descriptions come naturally with imprecisions, ambiguities, and subjectivities. The study of using natural language for information processing has been done for decades and many problems are so insurmountable that very little natural language processing is done in computer applications. Why then do we suggest the use of natural language for describing multimedia data? Our answer lies in the fact that everyone can describe multimedia data with natural language and that we believe that there are ways to limit the scope of the application of the natural language for descriptions as well as the fact that the scope of the language used in specific applications is generally is limited. Moreover, although we seem to suggest in the last section that natural language description is stored along with the media data, in reality, we actually employ a *parser* to convert the descriptions into more precise forms for processing. The parser is a separate and independent component of the system and advances in AI techniques would allow us to modify and enhance the parser capabilities as they become available. In this section we will discuss in more detail about our approach concerning this part of the system.

It should be pointed out that we have thought of the use of the simpler technique of forming descriptions with the use of keywords as in information retrieval. We discarded this approach because it is too imprecise for users to define contents of media data and the formulation of queries. Further, and more importantly, it is not always possible to convey exact meanings using only keywords. Much of the semantics are not expressible by this technique.

It is our belief that unrestricted natural language processing would be very difficult. The scope in this case is simply too broad. Fortunately we believe that each application naturally restricts the scope of descriptions. For example, if the application is to store the images of ships and aircrafts for identification purpose, one would not encounter descriptions like "flowers blooming" or "a dog chasing a cat". This natural restrictive phenomenon occurs in the non-multimedia data processing as well, and users have taken advantage of it in designing their databases and applications.

For each application a small set of vocabularies are likely to be the words used for description even if the system would not place any restriction on the users. This has been verified in many of the expert system applications. However, for a given media object, there are many words that would carry the same meaning and many ways one can say the same thing. We must find ways to let the system know when this happens. The parser is the component to handle this task and the *dictionary* or *lexicon* is where information is deposited for such a purpose. First let us say a few words what the parser output is going to be.

After studying the various ways of representing information, we have chosen the use of *first order predicate calculus* as a formal representation of the description data, which we receive from the user as text. Although there are a large variety of methods available to do this [Al87], e.g. semantic nets, frame-based methods, etc., we have not found significant advantages of using them. On the contrary, it seems to us that, when these methods are enhanced to become more formal or theoretically structured and sound, they tend to become very much similar to logic [So88]. At the same time, first order predicate calculus has the power of the other methods. Predicates can also be manipulated with programming languages like PROLOG, and they also allow for the easy development of storage structures and access paths, as we shall show later. This problem is really a knowledge representation problem and there is no pat solution at this time. We think predicate calculus representation is quite appropriate.

Thus, the parser translates the text description into a set of predicates. The details on the parser are beyond the scope of this paper and are given in a companion paper [MLR89]. The parser, however depends on the dictionary or the lexicon to turn descriptions into predicates. For every word used in an application, the dictionary names the grammatical class (e.g. noun, verb, etc.) and the templates of the predicates associated with that word. The templates contain slots to be filled with object identifiers or variables. Due to lexical ambiguity, there can be several entries for one word. For example, "can" might be a noun as well as a verb. In addition, many words may have the same meaning in an application. For example, "territory" and "region" may have the same meaning and both should be translated to "area" for a particular application. This fact would be defined in the dictionary. It is the parser's task with the use of the dictionary to resolve synonyms and to check the syntactical context to either resolve these ambiguities or to produce alternative interpretations [Al87].

Let us now discuss further about the predicates, each of which state a fact about the real world objects involved in the media object. An example of a predicate is "action (x, sign)". Real world objects and activities are referred to through their name, or identifier. The predicate states their properties - as indicated by the media object - and their relationships. In many cases, the name of the object may not be known, so that artificial identifiers have to be created. For instance, an image showing a car can be described by the predicates: car (x), manufacturer (x, Horch), year-built (x, 1922) to designate that the car was manufactured by Horch in the year 1922. The use of the name, x, connects different predicates that state the properties of the same object.

In our example application of the newspaper archive, the description phrase "Reagan and Gorbachev signing a treaty" is translated into a list of predicates like "conjunction (x1, Reagan, Gorbachev), action (x2, sign), agent (x2, x1), treaty (x3), object (x2, x3)." The next

phrase, "the event occurred on December 7, 1987", refers to the first phrase and therefore uses the identifier already introduced for the action of signing: "action (x4, occur), agent (x4, x2), date (x4, 07DEC87)." The same happens to the last phrase, "it took place in Washington, D.C.": "action (x5, take-place), agent (x5, x2), place (x5, 'Washington, D.C.')."

Choosing the right set of predicates is, of course, a very difficult task, comparable to knowledge acquisition for expert systems [Ga87]. For the purpose of this paper, it is sufficient to assume that the dictionary lists all the words the parser can recognize, all the parts of speech associated with any word, and the predicates to use when a word appears in the description. Thus, the set of all predicates that can be used in the descriptions must be defined in the dictionary.

For each media object, the description data is converted into a set of predicates interconnected by object identifiers. A query is also entered in textual, natural language description and is parsed by the parser. In contrast to the description, the arguments of the predicates can now be variables. A media object is selected for the result of the query, if and only if there exists a binding of those variables to object identifiers such that the description predicates of the media object *logically imply* all the query predicates. As an example, consider the following description:

> conjunction (x1, Reagan, Gorbachev), action (x2, sign), agent (x2, x1),
> treaty (INFtreaty), object (x2, INFtreaty), date (x2, 07DEC87).

Asking for a photo of Reagan and Gorbachev signing any treaty results in the following query predicates:

> conjunction (O1, Reagan, Gorbachev), action (O2, sign), agent (O2, O1),
> treaty (T), object (O2, T).

A binding of O1 = x1, O2 = x2, and T = INFtreaty establishes the query as a logical implication of the description and thus retrieves the photo. Please note that the query does not specify the date. This defines a particular model of search and retrieval that we will use as a basis of our discussion.

The matching discussed so far catches different natural language phrases with the same meaning, but it does not catch semantic relationships among predicates. If the description for an image is "a car with a red body", the predicates generated will be something like: car (x), component (x, y), body (y), color (y, red). A query that asks for "a red car" is translated into something like: car (X), color (X, red), and there would be no match. The system does not know that the color of a car's body is just the same as the color of the car.

To overcome this problem, *rules* can be introduced that express the semantic relationships among the predicates. In our example, the rule could be:

if car (A), component (A, B), body (B), color (B, C)
then color (A, C).

Using this rule, color (x, red) can be inferred in the example, and thus the query would match the description. This is similar to the use of S-rules in the START system [Ka88].

The definition of rules for the system and the parser to use is a powerful tool. This is absolutely necessary as there are always many ways one can describe same thing in natural language. In fact, the same person wanting to retrieve the same targeted items may likely state the queries differently at different times. Thus, instead of a query specifying "Reagan and Gorbachev signing a treaty", one may state the query as "Reagan signing a treaty in the presence of Gorbachev and vice versa". These two queries are syntactically not the same but there meanings are. Persons can see that these two queries are the same. Without the use of the rules to state that these are same, it is difficult at this state of the technology to leave it to the parser to detect the sameness of such queries, particularly the more complex ones. Further, this kind of research basically belongs to the domain of artificial intelligence, and a complete solution will not be forthcoming soon. On the other hand, our goal is to create a database system for handling media data. Our approach would allow us to make progress while permitting us to take advantage of other advances at the proper time by modifiying the parser when appropriate.

In addition to the rules and the words we have discussed, there is another aspect the system must handle. Just as words can mean the same thing, sometimes predicates may mean the same thing as well. For example, in this system, we want to permit users to enter predicates directly. In this case, one user may have a predicate of growth (x, tall) to describe an area with tall trees. Another user, or in fact, the same user at another time, may think of describing this as vegetation (x, tall). The dictionary must tell that both in fact means the same.

In short, the goal of the dictionary is to allow us to resolve all ambiguities, imprecisions and subjectivities. As such, the dictionary must be compiled by the database adminstrator for the application. It is then submitted to the database system as part of the database schema. The options as to how the dictionary is to be embedded in the database schema will be discussed in Section 5 when we look at the data structures.

4. Data Model Extension

To discuss the consequences of the use of descriptions and predicates in media object retrieval, we take a closer look at a possible embedding of different types of media into the *relational* database model. We have decided to use the relational model because of its flexibility and there is so much work done with this model, both practically as well as theoretically. Other good reasons can be found in [SR86, SR87].

We follow the arguments in [Ta80, Gr84] that media objects should be *attributes* of different types [LM88, MLW88]. Just as the standard data types have their set of operations like addition, subtraction, multiplication, and division for numbers, and concactenation, substring, and length for character strings, the new data types will have their specific operations that create and manipulate values. For image we have operators that retrieve components like height, width, pixel depth, and the colormap, and others that cut out portions or perform bitmap operations. For sound, the list of operations includes access to the sampling rate, the resolution, the compression method used, and the actual samples. Other operations record and play sound, cut out pieces of existing recordings, or concatenate two recordings.

The description is always attached to the media object; therefore, we have made it part of the attribute value (compare Figure 1). A few additional operators allow to modify and read the description:

```
UPDATE PHOTO_RELATION
SET PHOTO = ADD_DESCRIPTION (PHOTO, "Reagan and Gorbachev signing
                                the INFtreaty on Dec. 7th, 1987")
WHERE ARCHIVE_NO = 35763;
```

The same operator is also available on a sound attribute. The relation used in the query is defined as follows:

```
CREATE TABLE PHOTO_RELATION (ARCHIVE_NO integer4,
                             PHOTO IMAGE);
```

The most important operator for retrieval is the one that compares the description with a search phrase and returns a boolean value. Referring again to the example of the newspaper archive, the query could be formulated in a query language like SQL, extended with the operators on the data type image:

```
SELECT ARCHIVE_NO, HEIGHT (PHOTO), COLORMAP (PHOTO), ...
FROM PHOTO_RELATION
WHERE SHOWS (PHOTO, "Reagan and Gorbachev signing a treaty");
```

The equivalent operator for sound has a different name (TALKS_ABOUT) but otherwise does

exactly the same. The implementation of the SHOWS operator includes the parsing of the query phrase and the matching with the descriptions assigned to the PHOTO attributes in the PHOTO_RELATION.

The important point about this interface is that the users do not need to know the predicates at all. To them, the matching of the descriptions and the queries looks like some kind of "magic", because it involves a certain degree of understanding. Since there is no need to keep the original descriptions, once they have been translated into predicates, the operator that displays the description of a given media object will return a description generated from the predicates. Users will realize that they are different from the originals, but the meaning will be the same. This is pretty much the way human beings would report something they have heard. Like them the system tries to "understand" the descriptions instead of simply recording them.

Other operations have been defined for various purposes. A complete description of them is beyond the scope of this paper. However, we shall specify a subset for illustration purpose. (A more detailed description for the operations on images is given in [MLW88].) First, for clarity, let us discuss briefly how one can use the data type IMAGE in a database. The simplest way of assigning an image to an object is a schema like

OBJECT (O_ID, ..., O_IMAGE)

where OBJECT is the name of the relation (e.g. PERSON) followed by a list of attributes. O_ID is the object identifier. This structure is useful when each object instance has one single image. For an object instance that would have many images (e.g. a person may have many photographs), one can define the schema as

OBJECT (O_ID, ...)
OBJECT_IMAGE (O_ID, O_IMAGE)

In the OBJECT_IMAGE relation, O_ID and O_IMAGE will both be taken together to serve as a key. In the case where one tuple in a relation may have many images and one image may belong to many tuples (e.g. a database of cars where each car may appear in many pictures and a picture may contain many different cars), we need a schema as follows:

OBJECT (O_ID, ...)
IMAGE_OBJECT (I_ID, I_IMAGE)
IS_SHOWN_ON (O_ID, I_ID, ...)

The three types of schemas are shown in Figure 3.

To find all images of an object in schema 3, a join is needed. The join can be expressed as

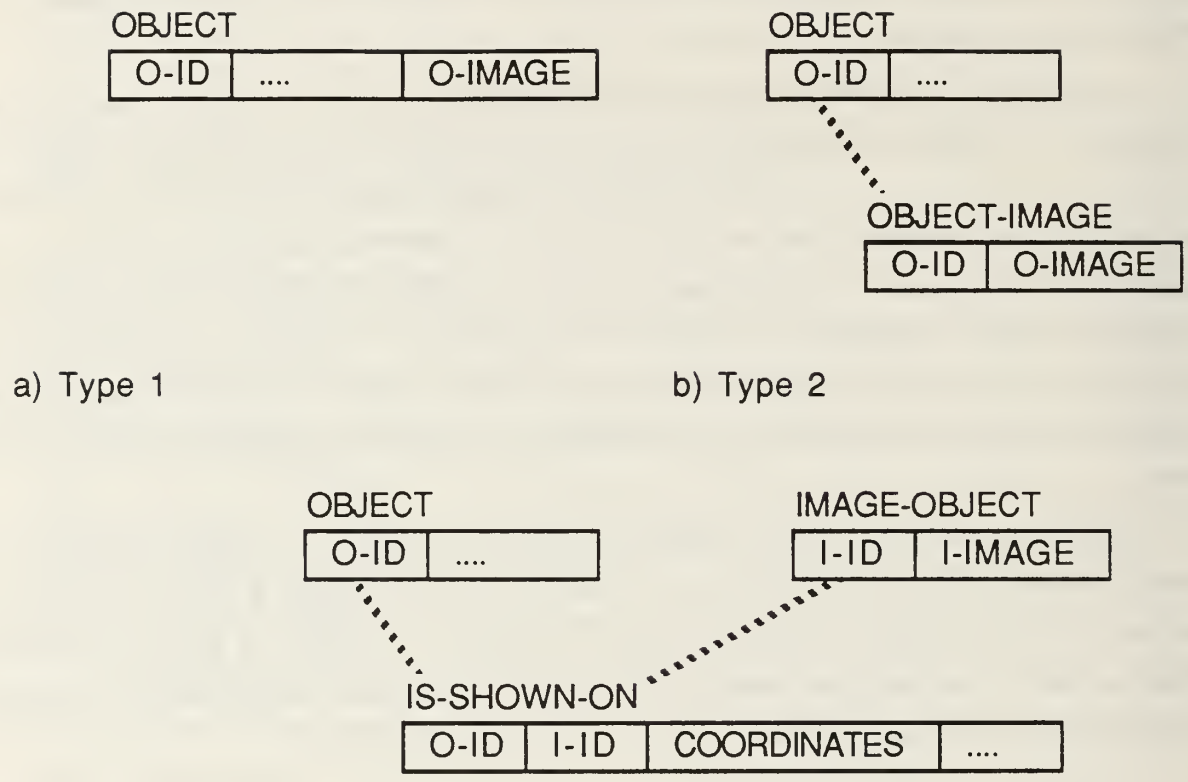

Figure 3: Three Schema Types to Model Relationships Between Objects and Images

NATJOIN (SELECT $_{O_ID=object1}$ (IS_SHOWN_ON), IMAGE_OBJECT)

To start with an image and retrieve the depicted objects, we have

NATJOIN (OBJECT, SELECT $_{I_ID=image1}$ (IS_SHOWN_ON))

One can even define a window on the image, use it to restrict the coordinates, and thus retrieve only the objects shown in the window. Note that, if the images appear in different relations (e.g. a relation of cars, another of ships, and still another of trains, and the image may contain objects in any one or more of these relations), then we must define three different IS_SHOWN_ON relations for this purpose.

We should note that the abstract data type operations are functions. For example, the function

CONSTRUCT_IMAGE (height, width, pixel-depth, encoding, colormap-length, colormap-depth, colormap, pixel-matrix)

produces a transient value of type IMAGE that cannot be assigned to program variables, but can only be used in INSERT and UPDATE statements of the query language. Thus we can have, for instance,

```
UPDATE IMAGE_OBJECT
SET I_IMAGE = CONSTRUCT_IMAGE ($height, $width, $depth,
                               RGB_COLORMAP, 256, ...)
WHERE I_ID = 1234;

INSERT (4567, CONSTRUCT_IMAGE ($ht, $wd, 24, IHS_NO_COLORMAP, 0, ...))
INTO IMAGE_OBJECT;
```

where $ represents program variables and parameters with capital letters only indicate named constants.

Retrieving attribute values of type IMAGE from the database into program variables uses another set of functions like:

```
HEIGHT (IMAGE attribute) : integer;
WIDTH (IMAGE attribute) : integer;
DEPTH (IMAGE attribute) : integer;
ENCODING (IMAGE attribute) : encoding-type;
    etc.
```

Each function has a specific output type. Different functions can be defined to produce different output types for an IMAGE attribute. A query using these functions may look like this:

```
SELECT HEIGHT (I_IMAGE), WIDTH (I_IMAGE), DEPTH (I_IMAGE)
INTO $yrange, $xrange, $dep
FROM IMAGE_OBJECT
WHERE I_ID = 33;
```

Other functions have been planned for various purposes. The list of functions cannot be complete until such a system is used in applications. More functions, however, can be added any time as needed.

Similar to the IMAGE data type, operations for the SOUND data type have been defined. Again, the list is by no means complete. We expect others will be added when needed. The

above however, can serve to indicate how the abstract data type operations on media data are defined and how they are incorporated into the database languages.

The operations to store and manipulate the descriptions associated with a media object fit smoothly into the list of operations. They are the same for IMAGE, SOUND, and any other media data type. The parser is invoked internally as part of the implementation of these operations. In the case where the users are experts, they might want to operate on the predicates directly instead of the textual descriptions. An additional set of operators on the media data type allows them to do so:

```
UPDATE PHOTO_RELATION
SET PHOTO = ADD_PREDICATES (PHOTO, "conjunction (x1, Reagan, Gorbachev),
                    action (x2, sign), agent (x2, x1), treaty (INFtreaty),
                    object (x2, INFtreaty), date (x2, 07DEC87),
                    place (x2, 'Washington, D.C.')")
WHERE ARCHIVE_NO = 35763;

SELECT ARCHIVE_NO, HEIGHT (PHOTO), COLORMAP (PHOTO), ...
INTO $number, $ht, $cm, ...
FROM PHOTO_RELATION
WHERE IMPLIES (PHOTO, "conjunction (X1, Reagan, Gorbachev), action (X2, sign),
                    agent (X2, X1), treaty (X3), object (X2, X3)");
```

The syntax of these SQL extensions is chosen for illustration purpose and is far from being completely and firmly defined. However, they should serve to illustrate the concept and the functionality of the system we have in mind.

5. Additional Data Structures

To support such a system efficiently, additional data structures must be defined. We must find methods of defining access paths conncecting the predicates and the data so that the media data can be searched through the predicates. We must also consider the organization of the dictionary and the rules for the parser and the system to use. We shall discuss these aspects in this section.

5.1. The Description Predicates

As we said earlier, each media object, i.e., each attribute value of a media data type, has a set of description predicates attached to it. Just like text this set of predicates is only loosely structured: it has variable length and a flexible structure. The sequence of the predicates is irrelevant and can be changed by the system. Since the users do not know very much about the predicates anyway, the database management system has a significant degree of freedom in organizing them. This is particularly important because they will be frequently used in search.

It is an open research issue how an access path for the set of sets of predicates belonging to one media attribute can be designed. One rather simple idea would be to add the internal identifier of the media object to the parameter list of the predicates: car (image2, x) will now state that the object x on the image2 is a car. This means that the predicates of all attribute values can be mixed, i.e., we form the union of all description predicate sets. The query predicates will also have an additional parameter position that is occupied by a variable. Regarding the query as a theorem, if it can be proven to be true, the binding of this variable will identify the media object. This can be implemented directly in PROLOG:

```
car (image1, x).
color (image1, x, green).
manufacturer (image1, x, "Horch").
year_built (image1, x, 1922).

query(I) :- car (I, Z), manufacturer (I, Z, "Horch").

?- query(Image).

yes.
Image = image1
```

As usual, words starting with capital letters indicate variables, whereas lower case words and strings in double quotes are object identifiers. Of course, this simply shifts the responsibility for the efficient search and matching over to the PROLOG runtime system.

Another way of organizing the predicates is to use a similar technique as we do in secondary indexing methods. A structure is created so that each predicate will have with it the list of media data that have this predicate value. For example, we will have

car : image1, image2, ...
color(,green) : imageI1, imageI2, ...
manufacturer(,Horch) : imageJ1, imageJ2, ...
etc.

In certain cases a predicate may generate entries in more than one list. For example, if the predicate of action (Reagan,sign) is used to describe the image "imageK1", the image identifier will show up in two lists:

action(Reagan,) : imageK1, imageK2, ...
...
action(,sign) : imageK1, imageL1, imageL2, ...

The first list will have all the images that contain Reagan doing something in the description and the second all the images with action "sign". In essence, if a predicate can have more than one value for each parameter position, each value will generate a separate and distinct list.

From the way predicates are generated from natural language descriptions, there are many predicates that will have no meaningful value in searches. For example, the words "the", "a", etc. have their places in natural language but do not give much information for searching. They are mapped to predicates like "definite (x)" and "indefinite (x)", and are used to resolve anaphoric references [Al87]: given "a red car in front of a house" and "the car has two doors", the object identifier created for the car of the first phrase must also be used for the car in the second phrase. After that, the predicates "definite", "indefinite", etc. are no longer needed. As is the practice in information retrieval in which "stop lists" are used to eliminate such words from appearing as keywords, we propose to use the same technique to eliminate these predicates from generating the indexing lists. This task again is the responsibility of the database adminstrator.

To use these lists for searching, we only need to receive the predicates generated by the parser on the query and proceed to retrieve those lists that have the corresponding predicates and parameter values. Intersection of the lists will give us the likely targeted items for retrieval. Naturally the predicates from the query would be stripped of those "stop-list" predicates before the system would proceed to process. This technique is simply the usual way of using secondary indexing in normal DBMS. However, as in the case of indexing non-root attributes in a hierarchical structures in the normal DBMS, *false drops* will occur. The system must therefore take the result of the intersection from the lists and examine each item in turn to verify that it does satisfy the query, if false drop is to be eliminated.

The number of the lists is expected to be very large even for a small application, because there are many predicates, and they may have many parameter values. To have an efficient search one has to consider how to organize the lists for easy accessing. First, we observe that, unlike the secondary index values in the normal DBMS where range values are meaningful, the predicates that generate the indexing lists do not have that property. This means that we can make use of *hashing* on the predicates to get to the lists. This makes accessing the lists fast and convenient. Each list would then be treated as the data that comes with the combination of predicate and parameter value treated as the key to be hashed. No additional accessing path would be needed for this part of the structure, namely organizing predicates for query search, and no better accessing method has been found at this time.

A detailed view into more appropriate methods can be found in the companion paper [MLR89].

5.2. The Lexicon (Dictionary) and the Rules

There are several options as to how the dictionary can be embedded in the database schema:

- one dictionary for the whole database
- one dictionary for each application (which here means defining
  a subset of the database as done in subschema definition.)
- one dictionary for every media type (IMAGE, SOUND, ...)
- one dictionary for every media attribute (e.g. I_IMAGE, PHOTO, ...)
- combinations of all

The first solution leads to problems if the same word has different meanings in different contexts. It also slows down the parsing process, because the parser always has to search the whole dictionary, although it is only using a subset. Having as many dictionaries as there are attributes will on the other hand introduce a significant amount of redundancy, because common words like "the", "and" and "not" have to be defined in all of them. Hence, the best approach would be to use several dictionaries: a common dictionary for the whole database with words like "the", "and", "not", etc., a more specific dictionary for each of the different applications, an IMAGE dictionary with words like "foreground", "background", "behind", "left", "right", etc., and an X_RAY_PHOTO dictionary with predicates for "skull", "bone", etc. In this scenario, the parser uses a combination of four dictionaries to translate a specific description.

The storage structure of the dictionary is yet to be developed. It caters mainly to the parser and hence should be optimized to support the parsers information needs. Right now, the parser always searches for natural-language words and retrieves the grammatical class and the predicate templates. Tree structures or hashing schemes could be used to support this search. They can utilize the assumption that the dictionaries are hardly ever modified. When generation of natural language from predicates is required, the dictionary will also be used in an inverted mode. It is our belief that access organization in this part is not a major problem, although different organizations will influence the parser's performance. Investigation into this part's data organization is left for solution at a later date.

The *rules* are associated with the dictionary. They could be used to extend all the descriptions in the database by additional predicates that can be inferred, but that would lead to a large storage consumption. They could also be used "backwards" to modify the query, which seems to be more appropriate. The details of the data structure for the rules are yet to be worked out. We should note, however, that rules are used in many expert systems and we believe that we can find one of their organizations would be satisfactory for our use as well. In fact, if our experience shows that there are not many rules for any given application, a purely sequential search of the rules may be quite satisfactory, as we see that rules are application dependent and each application will have its own set of rules.

As mentioned before, the lexicon and the rules must be defined by the database administrator. The syntax of the data definition language must be extended to allow for this definition. The result might look like this:

```
create media dictionary (lexicon: filename1,
                         rules: filename2);
create NEWSPAPER dictionary (lexicon: filename3,
                         rules: filename4);
/* NEWSPAPER is defined as an application or subschema elsewhere */
create IMAGE dictionary (lexicon: filename5,
                         rules: filename6);
create TEXT dictionary (lexicon: filename7,
                         rules: filename8);
create table PHOTO_RELATION (ARCHIVE_NO integer4,
                         PHOTO IMAGE (lexicon: filename9,
                                      rules: filename10));
```

Although the syntax is not yet firm, the above syntax shows how the dictionaries and rules are linked to the different levels: All media objects, media objects of an application, media

objects of a certain type (image, text, etc.), and finally media objects belonging to the same attribute. Each level can define new predicates. The rules can refer to predicates of their own level and to the predicates on higher levels.

6. Architecture

In this section we will identify the components of an MDBMS that actually deal with the data structures introduced in the previous section. From what we have mentioned so far it is clear that we need building blocks like:

conventional data management
media object management
description management
parser
language generator
matcher
query processing

The description management could be integrated with the conventional data management (e.g. in extensible database management systems like EXODUS [Ca86], Genesis [Ba86], or Starburst [Ha88]), but for now we keep them apart to point out their specific roles. Both rely on a lower level storage manager that takes care of things like file allocation and buffer management. The description management organizes the descriptions by relation and attribute, and each description will be linked to its media object and the other attributes of the same tuple by means of a tuple identifier (TID) or a surrogate.

The way how these building blocks interact is depicted in Figure 4. The query processing accepts queries from the users (sometimes represented by programs) and executes them by calling the other components. What each component is supposed to do is best shown by sketching the processing of different types of queries:

- Insert and Update

The query processing calls the conventional data management that takes care of the attributes with standard types, and the media object management components for all the media objects that are affected. The conventional data management selects a surrogate for the new tuple or retrieves the surrogate of an existing one. For every new description entered, the query processing calls the parser. The parser uses the dictionary to produce the predicates and returns them to the query processing. The query processing finally hands the predicates

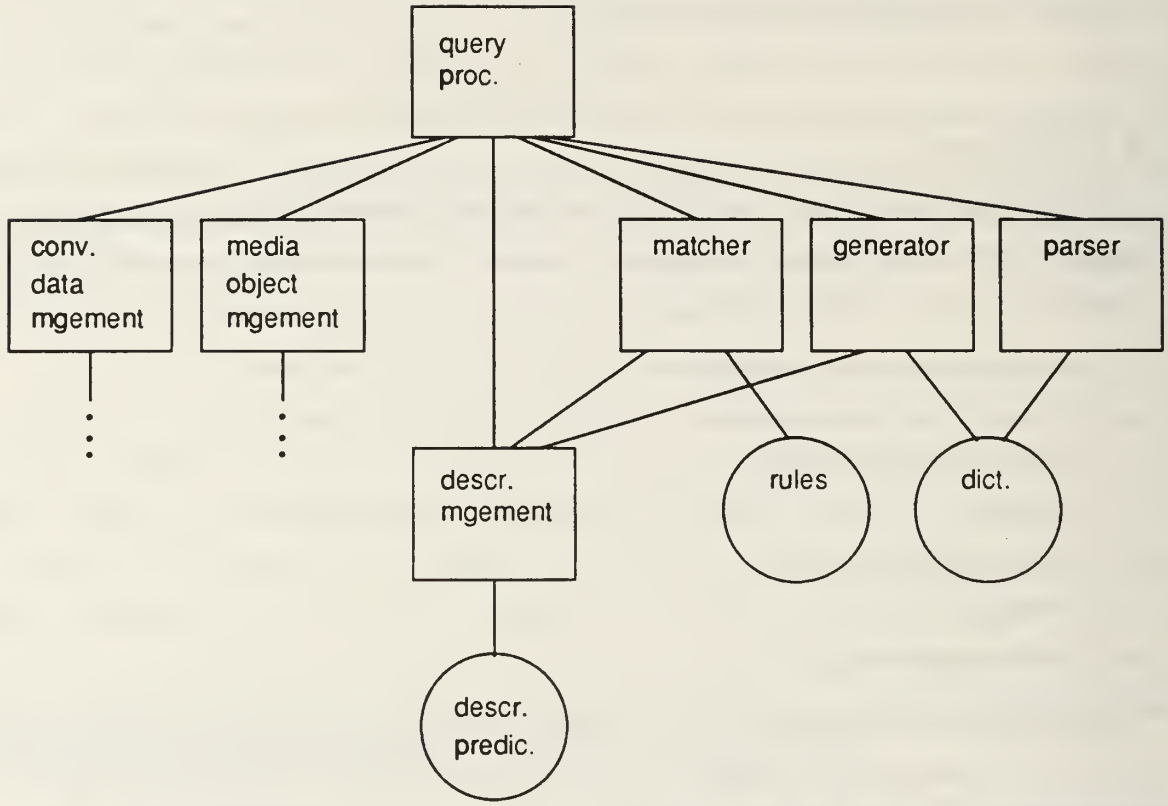

Figure 4: Proposed Architecture of an MDBMS: Building Blocks and Their Interaction

over to the description management, specifying relation name, attribute name, and surrogate.

- Query

The query processing decomposes a query into subqueries that can be handled by the conventional data management, the media object manager, or the description management separately. For the subqueries that only use the text descriptions, the query management calls the parser to obtain the query predicates. They are then handed to the matcher. The matcher calls the description management and also uses the rules. It returns a (possibly empty) list of surrogates to the query processing. For queries that specify the predicates directly, the parser is bypassed and the matcher is called immediately.

- Output

The query processing must provide the surrogates of the tuples to be presented. These surrogates are handed to the generator which calls the description management to get the predicates. It uses the dictionary to generate a sequence of natural language phrases, which it returns to the query processing for output.

The primary user interface we have in mind for the system is the one based on natural language for descriptions. As we have mentioned already, it might be useful to offer another interface for the expert who can work with the predicates directly. But we may also build applications on top of the predicate-oriented interface that again provide a different view to the end user.

For instance, a menu-driven approach could be chosen to enter descriptions. To avoid the ambiguity of natural language and also the undesirable situation that the parser rejects a description because it cannot find a word in the dictionary, the menu allows the user the choice among the available predicates using extensive explanations about what the choice means. For instance, in aerial photography the objects that can be recognized on an image are classified into regions (areas), lines, and point objects. Each of them has a huge variety of subtypes and properties: A region has variation in altitude, texture, vegetation, color, etc. A line can be a border of a region, a road, a power line, etc. and it has a direction, a shape, and a width. Point objects are telegraph poles, buildings, crossings, junctions. All objects may participate in relationships like adjacent, north-of, inside, etc.

A menu-driven system would ask the user to identify objects on an aerial photograph, to select their subtypes and properties, and finally to establish the relationships among them. Such a system can guarantee the consistent use of terms. Ideally, it would get all its domain-specific information from the dictionary. It is probably too difficult to produce a domain-independent menu system like that. More likely is the solution where the administrator who designs the dictionary also implements or at least specifies the menu interface. The knowledge about the meaning of the predicates becomes part of the code.

Another possible interface would allow the entry of predicates that need not be included in the predicate set defined in the dictionary. The syntax of the entered descriptions would be that of predicates, but the predicate names would be treated as natural-language words. This relaxes the need to know the dictionary in all its details. Assume that the dictionary declares a predicate "region (x)", that is used as the standard predicate for words like area, etc. Then a predicate "aera (z)" entered through this interface would still be accepted, although it is not one of the "standard" predicates defined in the dictionary. It would be transformed into "region (z)" using the information in the dictionary. The component that does this is called the predicate transformer. In essence, this approach is to allow users to define predicates

themselves in an impromptu mode instead of leaving this task to the DB adminstrator.

The query processing must use the predicate transformer whenever an expert user enters a query that specifies predicates, and the transformation must take place before anything is is done with the predicates, i.e. passing them to the description management or to the matcher.

Putting all the building blocks together we end up with the architecture diagram of Figure 5. It offers a choice of three different interfaces to work with the descriptions attached to media objects.

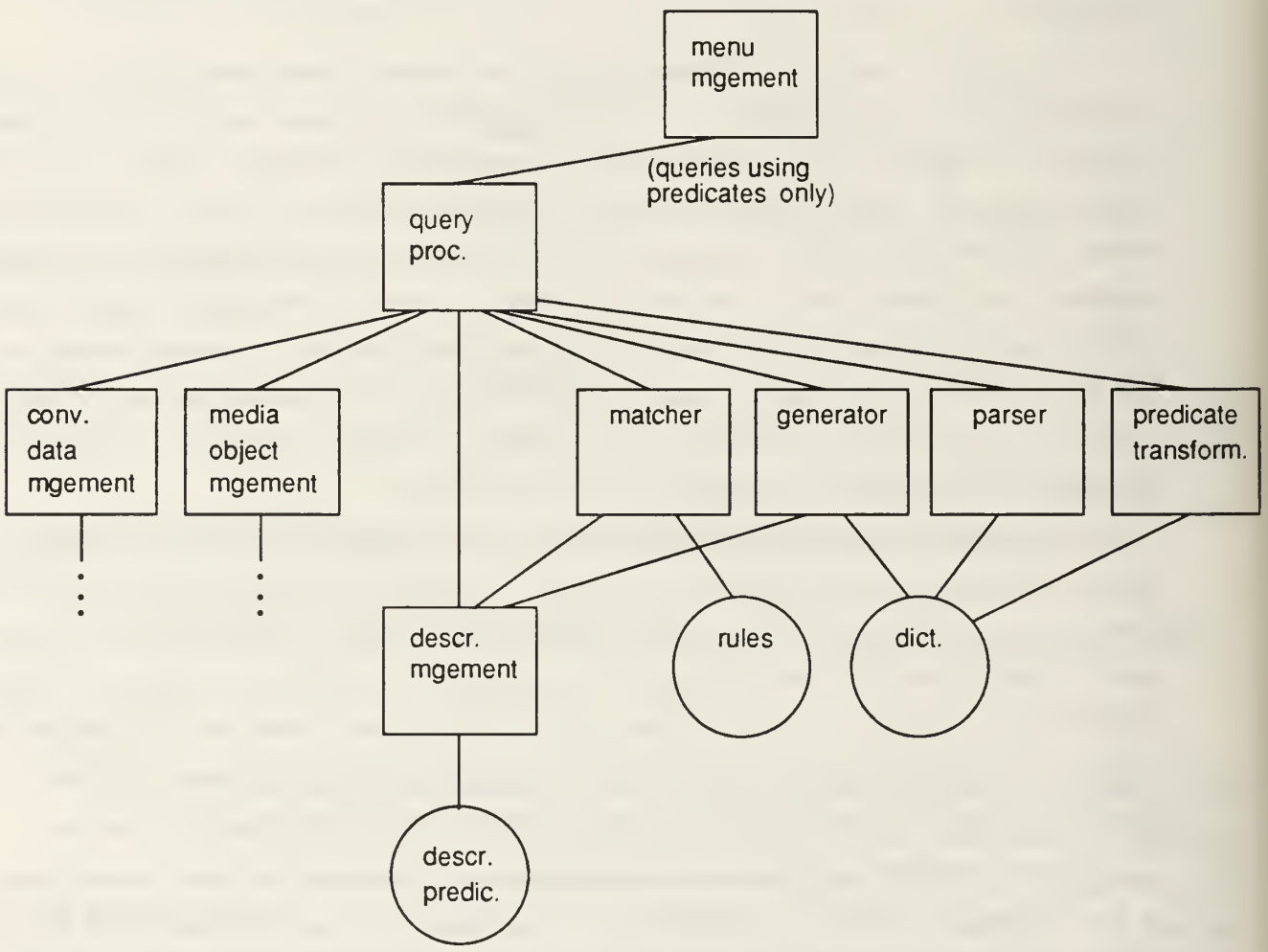

Figure 5: Complete Architecture of Proposed MDBMS

7. Conclusion

Handling multimedia data imposes a new challenge on database management systems. Despite many differences in data model and implementation aspects, all the research projects on Multimedia Database Management Systems (MDBMS) have decided to organize media data in new (abstract) data types. This is generally accepted as the adequate approach. However, none of the other projects have addressed the problem of content description and search of media data. We viewed this to be of prime importance for the widespread use of MDBMS and therefore concentrated on this aspect.

Text descriptions are attached to media data instances to capture their contents. This is very convenient for users, because the descriptions are easy to enter and easy to understand. However, they are not as easy to handle by the MDBMS. To reduce the vagueness and ambiguity involved with natural language, a parser is used to reduce the noun phrases and sentences to predicate lists, one might say: to condense and to give focus to the meaning of the text descriptions. Queries are entered as text phrases and are translated into predicate lists, too. The query evaluation has the flavor of theorem proving: A binding of variables must be found so that the description predicates of a media object logically imply the query predicates. If the binding can be found, the media object that satisies the query's specification will result.

Normal users work with the text descriptions only and do not care about the predicates used in the internal matching. Expert users may be allowed to see and manipulate the predicates directly. A third user interface has been proposed that lies in between: The unskilled user selects the predicates with the help of a sophisticated menu interface. There is no need for parsing, but the user still need not remember names and meaning of all the predicates allowed in this application.

The requirements of managing contents information in addition to the media data leads to a new architecture for MDBMS. It consists of basic building blocks for conventional data management, media data management, description (predicate list) management, parser, language generator, matcher, and query processing. We have shown the task of each building block and the ways how they interact with each other to process queries that involve multimedia data. This architecture provides a framework for the development of MDBMS that enables content search on multimedia data. The paper also indicates that to support the kind of processing needed, additional data structures must be constructed. The forming of indexes based on the predicates and their values is one such example. The architecture and the approach to solving the problem give rise to many interesting issues that need further attention and development effort.

The performance of the parser relies on the design of the dictionary. Our experience has shown us that much damage can result with a bad design. Whether guidelines can be given to produce a reasonably good design is an open issue at this time. Ideally, the system would even help in the design in the dictionary. Since the users enter natural language descriptions, it would be nice to generate at least an initial set of predicates automatically from those descriptions. This is much like the START system [Ka88], and its T-expressions could easily be transformed into predicates.

Another important aspect is performance analysis and quality assessment. There are two sources of imprecision in our approach. First, the natural-language description is necessarily subjective and may overlook or misconceive some interesting features. Second, the translation into predicates again ignores some of the more complex structures of natural language. It must be investigated how good retrieval can be done under these circumstances. Once this has been done, the next question is performance in terms of time: How fast can the retrieval be? This includes the evaluation of the access path structures we have suggested as well as the performance under real use. It is too early to have this kind of information.

Overall the paper has provided a novel approach to the handling of media data that allows us to perform content search on the images, sounds, graphics, etc. In some way it mimics the intelligent processing people do in their handling of these kinds of data. The approach is purposely structured in a modular manner, allowing us to apply AI techniques as progresses are made there. The application of the abstract data type concept permits us to develop operations for the different media as each medium becomes integrated into the computer. Although we have only shown some of these operations for image and few for sound, the reader should be able to see that operations can easily be specified for other media data in a similar manner.

Although a complete system has not been built, some major concepts have been explored and parts of the system have been implemented. The system can now store and retrieve image data in the manner as stated. However, the text descriptions are very restrictive. At this time, we are exploring the limit of purposely restricting the description data. This is an important aspect, because, as we stated earlier, unrestricted use of natural language is most likely destined to failure. We therefore are considering the adequacy of, for example, restricting the natural language description to only noun phrases. While this would not be able to describe all possible contents of media data, perhaps it would be enough to satisfy our need. In any case, such a capability in media data processing already far exceeds the capability of any system today.

To be sure, there are many other open problems, including data structure and database problems, as already stated in the paper. The overall concept and architecture, however, are

believed to be sound. Some of the details we have done are beyond the scope of this paper. Such is the case of the construction of the parser and the access path structure of the predicates. For those readers who are interested in this part of the work, please refer to our companion paper [MLR89] for details. In the future, we will continue to try to find solutions to the many problems and implement them into the system.

Aknowledgement

The authors are indebted to Neil Rowe for the design and implementation of the parser.

References

Al87 Allen, J., *Natural Language Understanding*, Benjamin/Cummings Publ. Co., Menlo Park, CA, 1987.

Ba86 Batory, D.S., Barnett, J.R., Garza, J.F., Smith, K.P., Tsukuda, K., Twichell, B.C., and Wise, T.E., "Genesis: A Reconfigurable Database Management System," University of Austin at Texas, Technical Report TR-86-07, March 1986 (also to appear in IEEE Trans. on Softw. Eng.).

Be85 Bertino, E., Gibbs, S., Rabitti, F., Thanos, C., and Tsichritzis, D., "Architecture of a Multimedia Document Server," in *Proc. 2nd ESPRIT Technical Week* (Brussels, Sept. 1985), 1985.

Be86 Bertino, E., Gibbs, S., Rabitti, F., Thanos, C., and Tsichritzis, D., "A Multimedia Document Server," in *Proc. 6th Japanese Advanced Database Symposium* (Tokyo, Aug. 1986), Information Processing Society of Japan, 1986, pp. 123-134.

BRG88 Bertino, E., Rabitti, F., and Gibbs, S., "Query Processing in a Multimedia Document System," *ACM Trans. on Office Information Systems*, vol. 6, no. 1, Jan. 1988, pp. 1-41.

Ca86 Carey, M.J., DeWitt, D.J., Frank, D., Graefe, G., Richardson, J.E., Shekita, E.J., and Muralikrishna, M., "The Architecture of the EXODUS Extensible DBMS," in *Proc. Int. Workshop on Object-Oriented Database Systems* (Pacific Grove, CA, Sept. 1986).

Ch86 Christodoulakis, S., Theodoridou, M., Ho, F., Papa, M., and Pathria, A., "Multimedia Document Presentation, Information Extraction, and Document Formation in MINOS: A Model and a System," *ACM Trans. on Office Information Systems*, vol. 4, no. 4, Oct. 1986, pp. 345-383.

Ga87 Gammack, J.G., "Different Techniques and Different Aspects on Declarative Knowledge," ch. 7 in Kidd, A.L. (ed.), *Knowledge Acquisition for Expert Systems*, Plenum Press, New York and London, 1987, pp. 137-163.

Gr84 Grosky, W.I., "Toward a Data Model for Integrated Pictorial Databases," *Computer Vision, Graphics, and Image Processing*, vol. 25, no. 3, March 1984, pp. 371-382.

Ha88 Haas, L.M., Cody, W.F., Freytag, J.C., Lapis, G., Lindsay, B.G., Lohman, G.M., Ono, K., and Pirahesh, H., "An Extensible Processor for an Extended Relational Query Language," IBM Research Report RJ6182, San Jose, April 1988.

Ka88 Katz, B., "Using English for Indexing and Retrieval," in *Proc. RIAO 88* (Cambridge, MA, March 1988), pp. 314-332.

KKS87 Kosaka, K., Kajitani, K., and Satoh, M., "An Experimental Mixed-Object Database System," in *Proc. IEEE CS Office Automation Symposium* (Gaithersburg, MD, April 1987), IEEE CS Press, order no. 770, Washington 1987, pp. 57-66.

LM88 Lum, V.Y., and Meyer-Wegener, K., "A Conceptual Design for a Multimedia DBMS for Advanced Applications," report no. NPS52-88-025, Naval Postgraduate School, Monterey, CA, August 1988.

MLR89 Meyer-Wegener, K., Lum, V.Y., and Rowe, N.C., "Data Structures Supporting Content Search in Multimedia Databases," (in preparation).

MLW88 Meyer-Wegener, K., Lum, V.Y., and Wu, C.T., "Image Database Management in a Multimedia System," report no. NPS52-88-024, Naval Postgraduate School, Monterey, CA, August 1988, to appear in *IFIP TC-2 Working Conf. on Visual Database Systems* (Tokyo, Japan, April 1989).

So88 Sowa, J.F., "Knowledge Representation in Databases, Expert Systems, and Natural Language," in *Proc. IFIP WG2.6/WG2.8 Working Conf. on the Role of Artificial Intelligence in Databases and Information Systems* (Guangzhou, China, July 1988), eds. C.-H. Kung and R.A. Meersman, North-Holland Publishing Co., Amsterdam.

SR86 Stonebraker, M., and Rowe, L.A., "The Design of POSTGRES," in *Proc. ACM SIGMOD '86* (Washington, DC, May 1986), ed. C. Zaniolo, pp. 340-355.

SR87 Stonebraker, M., and Rowe, L.A., "The POSTGRES Papers," University of California at Berkeley, Technical Memorandum No. UCB/ERL M86/85, June 1987.

Ta80 Tang, G.Y., "A Logical Data Organization for the Integrated Database of Pictures and Alphanumerical Data," in *Proc. IEEE Workshop on Picture Data Description and Management* (Asilomar, CA, Aug. 1980), IEEE Computer Society, catalog no. 80CH1530-5, pp. 158-166.

WK87 Woelk, D., and Kim, W., "Multimedia Information Management in an Object-Oriented Database System," in *Proc. 13th Int. Conf. on VLDB* (Brighton, England, Sept. 1987), eds. P.M. Stocker and W. Kent, Morgan Kaufmann Publishers, Los Altos, CA, 1987, pp. 319-329.

DUDLEY KNOX LIBRARY
3 2768 00347508 8